An Algorithm for Mining Implicit Itemset Pairs based on Differences of Correlations

Tsuyoshi TANIGUCHI and Makoto HARAGUCHI

Division of Computer Science, Hokkaido University
N-14 W-9, Sapporo 060-0814, Japan
{tsuyoshi, makoto}@kb.ist.hokudai.ac.jp

Abstract. Given a transaction database as a global set of transactions and its local database obtained by some conditioning to the global one, we consider a pair of itemsets whose degrees of correlations are higher in the local database than in the global one. A problem of finding paired itemsets with high correlation in one database is known as Discovery of Correlation, and some algorithms to search for such characteristic paired itemsets are already proposed. However, even non-characteristic paired itemsets in the local database are also meaningful, provided the degree of correlation increases much higher in the local database than in the global one. They can be an implicit and hidden evidence showing that something particular to the local database occurs even though they are not yet realized as characteristic ones in the local. From this viewpoint, we have already proposed to measure the significance of paired itemsets by the difference of two correlations before and after the conditioning to the local database, and define a notion of DC pairs whose degrees of differences of correlations are high. As DC pairs are regarded as compound itemsets consisting of two component itemsets, we can have two basic strategies for finding them. One strategy firstly examines the compound itemsets and then the components, while another one does the component itemsets and then the compound ones. According to the former strategy, which we have already proposed and tested for its effectiveness, we have to enumerate many number of candidate compound itemsets that cannot be decomposable to components. For this reason, this paper presents a new algorithm according to the second strategy. It firstly enumerate possible component itemsets based on a new pruning rule for cutting off useless components. Secondly it forms the compound itemsets by combining the components thus detected, while we also make use of a constraint for preventing our algorithm from checking meaningless combinations.

1 Introduction

In the studies of data mining from transaction databases, many studies have been paying much attention to finding itemsets with high supports, paired itemsets appeared in association rules with high confidence [4], or paired itemsets with strong correlation [8–11]. These notions are considered useful for distinguishing characteristic paired itemsets with strong correlation in a single transaction database. A similar strategy based on the notion of change of supports, known as Emerging Patterns [5], is successful even for finding itemsets characterizing either of two databases. All of these notions about itemsets are thus proposed to extract paired itemsets required to be characteristic in a given database or either of two or more databases.

Although some users regard characteristic paired itemsets with strong correlation as useful, others may often regard many number of such paired itemsets as trivial because of the reason that they have been already known without examining a database. On the other hand, as is indicated in the study of Chance Discovery [12], some itemsets not characteristic in the above sense are also useful, as they are *potentially significant* under some condition.

For instance, suppose a database in which information about ages of customers and goods they purchased are stored. There may exist several pairs of particular ages and goods with high correlations, if people at those ages have a general tendency to buy those goods. In this case, the degree of correlation is not much dependent on time stamp data. As a result, there will be a little difference between the degree of correlation in the whole database and one in a local database of transactions with the recent time stamp data. On the other hand, there may exist another kinds of goods which teen-agers, for instance, begin to drastically buy just recently. As the purchase actions made by those aged people just begin, the overall degree of correlation between the ages and the goods is still low. However, its degree observed by restricting the transactions to those with the recent time stamp will show a significantly higher value.

Thus, the notion of potential significance we would like to define is the difference of degrees of correlation before and after some conditioning by which a local database is derived. Although we can consider various ways of conditioning and the corresponding local databases, we try to present a general algorithm to find a significant paired itemset with high change ratio of correlations, given a global and a local database. In section 6, a database with items designating places of transactions is examined. In this case, the conditioning is given by specifying particular place of transactions. The task of our algorithm is to find paired itemsets with higher correlation in the particular area, compared with the correlation in the whole area. Again it should be noted that the former correlation in the particular area need not be high, as we can interpret such a paired itemset as an implicit evidence showing that something particular to the local area occurs.

From the viewpoints mentioned in the above, we have already defined the notion of DC pairs and presented an algorithm to find them in [1]. More precisely, given a global and a local transaction databases, an itemset pair with higher change ratio of correlations is called a DC pair. A DC pair is syntactically regarded as a compound itemset consisting of two component itemsets. So, the algorithm presented in [1] is designed so that it firstly examines the compound itemsets and then the components, using two parameters for restricting the search spaces for the compound and the component itemsets. Although the algorithm is equipped with some pruning rules, an experimental result showed that large number of useless compound itemsets never decomposable into candidates in the space of component itemsets are generated and tested. Consequently, the subspace our algorithm actually visited turned out to be a very large one.

From the experimental observation thus obtained, in this paper, we present another new algorithm that enumerates component itemsets firstly and then combines those detected components into compound ones. It is clear from the definition that there exists no chance for the algorithm to examine any compound itemset not decomposable to possible component itemsets. Additionally, we can show that it suffices to check only transactions containing the candidates components in the local database in order to identify possible combinations of components. As the number of such transactions is not many, our algorithm can effectively generate compound itemsets from the set of candidate component itemsets. On the other hand, in the process of generating components, we can enjoy a monotone property over itemsets, depending on the parameters, that is also useful to prevent our algorithm from generating useless component itemsets.

Thus, in both processes of generating components and of combining them into compound itemsets, the number of generated candidates are restricted.

2 Related Works and Paper Organization

There exist many works in the field of data mining that are based on a strategy of contrasting two or more databases in order to extract significant properties or patterns from a huge data set. Particularly, data mining techniques, known as contrast-set mining [5–7], have been designed specifically to identify differences between databases to be contrasted.

For instance, in the study of Emerging Patterns [5] for two transaction databases, itemsets whose supports are significantly higher in one database than in another one are considered significant, as they can be candidate patterns for distinguishing the former from the latter. A similar strategy is also used in the system STUCCO [6] in order to obtain characteristic itemsets in one database based on χ^2 test. In addition, the system, Magnum Opus [7], examines relations between itemsets and a database among several databases. On the other hand, what this paper tries to find are paired itemsets whose correlations drastically increase in one database. Thus we can say that the subject of this paper is a kind of "contrast-set mining of correlations between itemsets".

Secondly, many methodologies have been proposed to detect characteristic correlations in a single database [8–10]. In these studies, using some function measuring the degree of correlation between itemsets, strongly correlated itemsets in a given database or in one database from given two databases are examined. Thus, these methods are also used to discover itemsets or family of itemsets that are characteristic in one database. On the other hand, the algorithm presented in this paper is designed so as to find even paired itemsets whose correlation in one database is not significantly high but is significantly higher than correlation in another database. Our algorithm may find the characteristic paired itemsets as special cases, but is never supposed to find only characteristic ones. To find these paired itemsets, we present some new pruning rules so that the algorithm successfully detects even non-characteristic paired itemsets.

Several notions about correlations have been proposed and used in the above previous studies from information theoretic or statistical viewpoints. If we need to consider even negative events $\overline{Y}$ that an itemset Y does not appear in transactions, the notion of correlations between two itemsets X and Y based on χ^2-test shall be taken into account. However, this paper is concerned with the notion of correlation in the sense that the number of chances for Y to occur increases under the presence of X. The degree of correlation in this sense can be calculated by the ratio $P(Y|X)/P(Y)$, known as self-mutual information by taking log.

Finally, we discuss the relation between a condition itemset which decide a local database and itemset pairs we try to find. If the condition is regarded as an antecedent of some rule, the itemset pairs can be considered a consequent of the rule. For example, association rule [4] is a rule whose consequent is an itemset such that the conditional probability of the itemset given by the antecedent (the condition in this paper) is no more than some parameter. That is, itemsets such that the probability of the itemsets is high in local database can be found by association rule. On the other hand, we find even itemsets such that the probability of the itemsets is very low in local database as a result of detecting high changes of correlation by the conditioning to the local database. Further, the correlation between the condition and the itemset pair is not always high. Briefly speaking, we try to find rules such that a consequent of the rule is an implicit itemset pair with high degree of change of correlations under an antecedent (a condition) of the rule.

The rest of this paper is organized as follows. The next section defines some terminologies used throughout this paper. In Section 4, we introduce the notion of DC pairs and define our problem of mining DC pairs. An algorithm for finding DC pairs is described in Section 5. Section 6 presents our experimental results. In the final section, we summarize our study and discuss future work.

3 Preliminaries

Let $\mathcal{I} = \{i_1, i_2, \cdots, i_n\}$ be a set of *items*. An *itemset* is a subset of $\mathcal{I}$. A *transaction database* $\mathcal{D}$ is a set of transactions, where a transaction is an itemset. We say that a transaction t *contains* an itemset X, if $X \subseteq t$. For a transaction database $\mathcal{D}$ and an itemset X, the *occurrence* of X over $\mathcal{D}$, denoted by $O(X, \mathcal{D})$, is defined as $O(X, \mathcal{D}) = \{t | t \in \mathcal{D} \wedge X \subseteq t\}$, and the *probability* of X over $\mathcal{D}$, denoted by $P(X)$, is defined as $P(X) = |O(X, \mathcal{D})|/|\mathcal{D}|$.

For an itemset C, a *sub-database* of $\mathcal{D}$ w.r.t. C, denoted by $\mathcal{D}_C$, is defined as the set of transactions containing C in $\mathcal{D}$, that is, $\mathcal{D}_C = O(C, \mathcal{D})$. The *complement* of $\mathcal{D}_C$ w.r.t. $\mathcal{D}$ is denoted by $\overline{\mathcal{D}_C}$ and is defined as $\overline{\mathcal{D}_C} = \mathcal{D} - \mathcal{D}_C$.

For itemsets X and Y, the *correlation* between X and Y over a transaction database $\mathcal{D}$, $correl(X, Y)$, is defined as $correl(X, Y) = P(X \cup Y)/P(X)P(Y)$. For a sub-database $\mathcal{D}_C$, the correlation between X and Y over $\mathcal{D}_C$, $correl_C(X, Y)$, is given by $correl_C(X, Y) = P(X \cup Y|C)/P(X|C)P(Y|C)$, where $P(X|C) =$

$P(X \cup C)/P(C)$. Note here that correlations are defined for only itemsets X whose supports in $\mathcal{D}$ and $\mathcal{D}_C$ are non-zero. We regard a pair of X and Y such that $correl(X, Y) > 1$ as characteristic since $P(X|Y) > P(X)$ holds. Note that $P(Y|X) > P(Y)$ holds, too. Similarly, we regard a pair of X and Y such that $correl(X, Y) \leq 1$ as non-characteristic.

4 DC Pair Mining Problem

In this section, we define a notion of DC pairs and our problem of mining them.

For a pair of itemsets X and Y, we especially focus on "difference of correlations observed by conditioning to a local database". Suppose here that an itemset C is a condition given by users. The difference of correlations is measured by the following ratio:

$$change(X, Y; C) = \frac{correl_C(X, Y)}{correl(X, Y)} = \frac{P(C)P(C|X \cup Y)}{P(C|X)P(C|Y)}. \quad (1)$$

Let $\rho(> 1)$ be an admissible degree of difference of correlations. In our framework, a pair of itemsets X and Y is considered significant if $change(X, Y; C) \geq \rho$ holds. Since we assume C is given by users, $P(C)$ can be regarded as a constant. Therefore, the difference is actually evaluated with the following function g:

$$g(X, Y; C) = \frac{P(C|X \cup Y)}{P(C|X)P(C|Y)}. \quad (2)$$

A pair of itemsets X and Y is called a *DC pair* if $g(X, Y; C) \geq \rho/P(C)$. We try to find all DC pairs efficiently. It should be noted here that the function g behaves *non-monotonically* according to expansion of itemsets X and Y. So we cannot apply a simple pruning method like one Apriori adopted [4]. Therefore, we approximate the above problem according to the following naive strategy:

> Find pairs of X and Y which give higher values of $P(C|X \cup Y)$, keeping the values of $P(C|X)$ and $P(C|Y)$ small.

In order to control the values of $P(C|X \cup Y)$, we use a new parameter ζ $(0 \leq \zeta \leq 1)$. Given ρ and ζ, we use a new parameter ϵ such that $\epsilon^2 = \zeta \cdot P(C)/\rho$ in order to control each value of $P(C|X)$ and $P(C|Y)$. Note here that ϵ can be replaced with another parameter if the value of $P(C|X)P(C|Y)$ is low.

Definition 1. DC Pairs Mining Problem
Let C be an itemset for conditioning. Given ρ and ζ, the DC pair mining problem is to find any pairs of X and Y such that $P(C|X \cup Y) > \zeta$, $P(C|X) < \epsilon$ and $P(C|Y) < \epsilon$, where $\epsilon = \sqrt{\zeta \cdot P(C)/\rho}$. We say that $X \cup Y$ is a *compound itemset* and X and Y are *component itemsets.*

5 An Algorithm for Finding DC Pairs

In this section, we present an algorithm to solve the DC pair mining problem. At first, we discuss a basic strategy of finding DC pairs. Next, we prove a pruning rule in order to find candidates for component itemsets efficiently. Finally, we show some constraints of DC pairs in order to restrict the combinations of the candidates properly.

5.1 A Basic Strategy of Finding DC Pairs

At first, we discuss a basic strategy of finding DC pairs. The DC pairs we try to find are pairs of itemsets X and Y such that $X \cup Y$ is a compound itemset and X and Y are component itemsets. Then, two strategies of finding DC pairs can be considered mainly. One strategy is that compound itemsets are identified and each compound itemset is divided into component itemsets. And another strategy is that component itemsets are identified and their compound itemsets are found. The former strategy has already been tried to find DC pairs in [1] and there exist some difficulties. In order to explain the difficulties, we show properties of $P(C|X)$ and DC pairs based on the following observation.

Consider an itemset X appeared in a global and a local databases. And note that $P(X) \neq 0$ and $P(C \cup X) \neq 0$ must hold. Also, as the size of X is longer, $P(X)$ and $P(C \cup X)$ tend to become lower. Since $P(C|X)$ is non-zero, $P(C|X)$ tend to be high in the case that $P(X)$ is low. That is, when the size of X is long, $P(C|X)$ tends to be high. This means that there exist a few candidates for component itemsets whose size is long in the database. Next, consider two itemsets X and Y and suppose here that the size of $X \cup Y$ is almost the same size of maximal transactions in the database. Since the size of either X or Y is necessarily no more than the half size of $X \cup Y$, either $P(C|X)$ or $P(C|Y)$ tend to be high. This means that there are a small number of DC pairs X and Y such that the size of $X \cup Y$ is long because either $P(C|X) < \epsilon$ or $P(C|Y) < \epsilon$ is difficult to hold. Therefore, there are many candidates for compound itemsets which cannot be divided into DC pairs in the database.

Based on the above observation, there is a difficulty of the strategy of finding candidates for compound itemsets. So, in this paper, we discuss the strategy of finding candidates for component itemsets in a bottom-up manner. Of course, if the number of the candidates components is large, the number of combinations of the candidate components is also very large. If the number of the candidates is N, the number of the combinations is $O(N^2)$. However, as the combinations can be restricted by using some constraint, it is expected that the number of the combinations is not so many.

After all, our strategy of finding DC pairs is that component itemsets are identified firstly, and the computation for mining DC pairs is divided into two phases:

Phase1: Identifying Component Itemsets

An itemset X such that $P(C|X) < \epsilon$ is identified as a candidate for a component itemset.

Phase2: Combining Component Itemsets

One component X is combined with another one Y such that $P(C|X \cup Y) > \zeta$.

5.2 Pruning Search Branches in Phase 1

In Section 4, by using parameters ζ and ϵ, we restrict DC pairs we try to find. Although $P(C|X)$ behaves *non-monotonically* according to expansion of an itemset X as well as g, we prove that a monotone property over itemsets can be observed depending on ϵ. In Phase 1, we consider a problem of mining candidates for component itemsets X in a bottom-up manner. During this search, we can prune useless branches (itemsets) based on the following observation.

Let X be an itemset and Z be an itemset containing X. Suppose that there exists a superset Z' of X such that $X \subseteq Z' \subseteq Z$ and $P(C|Z') < \epsilon$. Since $P(C|Z') = P(C)P(Z'|C)/P(Z') < \epsilon$, $P(C \cup Z') < \epsilon \cdot P(Z')$ holds. Therefore, $P(C \cup Z) \leq P(C \cup Z') < \epsilon \cdot P(Z') \leq \epsilon \cdot P(X)$. As the result, we have $P(C \cup Z) < \epsilon \cdot P(X)$. This means that if $P(C \cup Z) \geq \epsilon \cdot P(X)$ holds, then we cannot obtain any Z' such that $P(C|Z') < \epsilon$. That is, if $P(C \cup Z) \geq \epsilon \cdot P(X)$ holds, any Z' does not have to be examined.

Pruning Rule:
For a search node (itemset) X and a superset Z such that $X \subseteq Z' \subseteq Z$, if $P(C \cup Z) \geq \epsilon \cdot P(X)$, any Z' never be a candidate node of X in our search process.

When X examined in our bottom-up search can be applied to the above pruning rule, Z' do not have to be examined. However, there is a problem that the way of identifying a superset Z of X properly. We describe the way of identifying Z and a termination condition in Phase 1 in the next section.

5.3 Termination Condition in Phase1

In the previous section, we present a pruning rule in Phase 1. In order to use our pruning rule effectively, in sub-database $\mathcal{D}_{\mathcal{C}}$, while an itemset X is examined in a bottom-up manner, a superset Z of X have to be checked simultaneously. And, if the size of Z is long and our pruning rule can be applied to, many itemsets can be pruned. In order to identify such Z, the notion of look ahead in [2] can be used. Originally, the notion is used to find a frequent maximal itemset by checking a superset of an itemset examined. In order to use the notion, suppose a lexical ordering of the items and let X be an itemset examined at present. Let $tail(X)$ be the greatest item of X and $T(tail(X))$ be a set of a possible item which is greater than $tail(X)$ according to the lexical ordering. Then, X is expanded by adding an item $i \in T(tail(X))$ in order to avoid duplications. Note here that an itemset Z such that $Z = X \cup T(tail(X))$ is a potentially frequent maximal itemset which does not contain other itemsets in the database. That is, the size of Z is approximately the size of a maximal transaction and Z whose size is long is useful for pruning of many search nodes (itemsets). Further, although Z is not always a maximal itemset, we do not have to check whether Z is a maximal itemset or not. Rather, Z in this case is also useful for our search because a maximal itemset is not always able to be applied to the pruning rule and an itemset whose size is middle may be applied to. It should be noted that the cost of checking Z is not so high as only $\mathcal{D}_\mathcal{C}$ is examined.

Termination Condition of search in Phase 1:
For an itemset X and an itemset $Z = X \cup T(tail(X))$, if $P(C \cup Z) \geq \epsilon \cdot P(X)$, X is not expanded further in our search.

5.4 An Algorithm for Finding Candidates for Component Itemsets

We show a termination condition of our search in Phase 1 in the previous section. In this section, in order to implement the termination condition, we simply explain an algorithm for finding candidates for component itemsets.

At first, we use *backtracking* algorithm [2, 3] in order to enumerate candidates for component itemsets. Backtracking algorithm is based on recursive calls. Normally, an iteration of the algorithm inputs a frequent itemset F whose probability is no more than some parameter, and generates itemsets by adding every possible items to F. However, an iteration of our algorithm inputs an itemset X whose probability is non-zero because even itemsets whose probability is very low may be DC pairs and it is difficult to set a really proper parameter of probability. For each itemset whose probability is non-zero among itemsets generated, the iteration generates recursive calls with respect to it. To avoid duplications, an iteration of backtracking algorithms adds items contained $T(tail(X))$.

Next, when an itemset X is examined, a proper $T(tail(X))$ is not known yet. Let i be $tail(X)$, $\mathcal{D}$ be a global database and $\mathcal{D}_\mathcal{C}$ be its local database. Then, the probability of $X \cup \{j\}$ $(j \in T(tail(X - \{i\})), j > i)$ in $\mathcal{D}$ and $\mathcal{D}_\mathcal{C}$, and $X \cup T(tail(X - \{i\}))$ in $\mathcal{D}_\mathcal{C}$ have to be calculated. The probability of $X \cup T(tail(X - \{i\}))$ is calculated in order to check whether X fulfill the termination condition or not. On the other hand, the probability of $X \cup \{j\}$ is calculated in order to check whether j can be contained $T(tail(X))$ or not. That is, although we do not use any parameter of probability, an itemset whose probability is zero is, of course, trivial. Because the probability of a superset of $X \cup \{j\}$ is zero if the probability of $X \cup \{j\}$ is zero, j do not have to be added $T(tail(X))$. In order to calculate the probability of $X \cup \{j\}$ efficiently, the notion of *occurrence deliver* [3] can be used. Let $\{j_1, j_2, \cdots, j_m\}$ be $T(tail(X - \{i\}))$. Occurrence deliver computes the probability of $X \cup \{j_1\}, X \cup \{j_2\}, \cdots, X \cup \{j_m\}$ at once by tracing transactions containing X in $\mathcal{D}$ and $\mathcal{D}_\mathcal{C}$. It uses a bucket for $j_1, j_2, \cdots, j_m$, and set them to empty set at the beginning. Then, for each transaction t containing X, occurrence deliver inserts t to the bucket of $j_1, j_2, \cdots, j_m$. After these insertions, the bucket of $j_1, j_2, \cdots, j_m$ is equal to $O(X \cup \{j_1\}, \mathcal{D}_\mathcal{C}), O(X \cup \{j_2\}, \mathcal{D}_\mathcal{C}), \cdots, O(X \cup \{j_m\}, \mathcal{D}_\mathcal{C})$ if $\mathcal{D}_\mathcal{C}$ is examined.

Based on the above techniques, our algorithm for finding candidates for component itemsets is summarized as follows. In the algorithm, suppose that, for each item e such that $P(C \cup \{e\}) \neq 0$, $P(C \cup \{e\})$ and $P(\{e\})$ are calculated in advance. And let $T(tail(\{e\} - tail(\{e\})))$ be $T(tail(\{e\}))$. For each item e, by using the following algorithm, the candidates for component itemsets can be found.

```
ALGORITHM FindCandidateComponent(X)
IF $P(C \cup X)/P(X) < \epsilon$ then Output $X$;
$T(tail(X)) = \emptyset$;    $count\_look\_ahead = 0$;
For each item $i$ such that $i \in T(tail(X - \{tail(X)\}))$ do
  $Bucket_c[i] = \emptyset$;    $Bucket[i] = \emptyset$;
End for
For each transaction $t_c$ such that $t_c \in \mathcal{D}_\mathcal{C}$ and $X \subseteq t_c$ do
  IF $(C \cup X \cup T(tail(X - \{tail(X)\}))) \subseteq t_c$ then $count\_look\_ahead++$;
                                                  // look ahead
  For each item $j$ such that $j \in t_c$ and $j > tail(X)$ do
    Insert $t_c$ to $Bucket_c[j]$;                // occurrence deliver
    IF $j \notin T(tail(X))$ then $T(tail(X)) = T(tail(X)) \cup \{j\}$;
  End for
End for
IF $T(tail(X)) \neq \emptyset$ and $P(C \cup X \cup T(tail(X - \{tail(X)\}))) < \epsilon \cdot P(X)$
then                                              // our pruning rule
  For each transaction $t$ such that $t \in \overline{\mathcal{D}_\mathcal{C}}$ and $X \subseteq t$ do
                                                  //$\overline{\mathcal{D}_\mathcal{C}} = \mathcal{D} - \mathcal{D}_C$
    For each item $k$ such that $k \in t$ and $k \in T(tail(X))$ do
      Insert $t$ to $Bucket[k]$;                  // occurrence deliver
    End for
  End for
  For each item $e$ such that $e \in T(tail(X))$
    $O(X \cup \{e\}, \mathcal{D}_\mathcal{C}) = Bucket_c[e]$;
    $O(X \cup \{e\}, \mathcal{D}) = Bucket_c[e] + Bucket[e]$;
    IF $P(C \cup X \cup \{e\}) \neq 0$ then call FindCandidatePart($X \cup \{e\}$);
  End for
End if
```

5.5 Constraints of DC pairs in Phase2

After we find candidates for component itemsets in Phase 1, we have to combine one component with another one in order to find DC pairs finally. If the number of the candidates is large, the number of the combinations is very large. However, two constraints of DC pairs can be used in order to restrict the combination.

At first, we describe a basic constraint of DC pairs. The DC pairs are pairs of itemsets X and Y such that two itemsets do not overlap. Then, if both X and Y contain some same item, pairs of X and Y are not DC pairs. In this case, combined itemsets $X \cup Y$ do not have to be examined. Secondly, we explain a main constraint of DC pairs. If pairs of X and Y are DC pairs, $P(C \cup X \cup Y) \neq 0$ must hold. Therefore, Y is necessarily contained by transactions containing X

in a sub-database $\mathcal{D}_{\mathcal{C}}$. Also, $P(C \cup X)$ is low in many case if $P(C|X) < \epsilon$ holds. For example, if $\epsilon = 0.1$ and $P(X) = 0.5$, $P(C \cup X) < 0.05$. Therefore, we firstly check whether or not Y is contained by transactions which contain X in $\mathcal{D}_{\mathcal{C}}$, and if Y is not contained by the transaction, Y does not have to be examined. The combinations actually examined in detail are restricted properly by only checking the small number of such transactions.

6 An Experiment

In this section, we present our experimental results. The purpose of experiments is to confirm that DC pairs can be found efficiently by using our pruning rules and constraints, and potentially significant DC pairs can be actually found for a given database.

6.1 Dataset and Implementation

We conducted the experiments on *Entree Chicago Recommendation Data*, a database in the UCI KDD Archive [13]. It consists of eight databases each of which contains restaurant features in a region, e.g. Atlanta, Boston and so on in the USA. The eight databases are combined into a single database $\mathcal{D}$ referred to as the global one. With the conditioning by each region C, we define a local (sub-)database $\mathcal{D}_{\mathcal{C}}$ in $\mathcal{D}$. The global database consists of 4160 transactions each of which is a subset of 265 items, where each item represents a feature of restaurant, e.g., "Italian", "romantic", "parking" and so on. Thus, a transaction $\{f_1, f_2, f_3\}$ means there exists a restaurant with the feature f_1, f_2 and f_3.

Based on the algorithm presented in the previous section, our system has been implemented in C and run on a PC with 1.00 GB RAM and a Xeon 3.60 GHz processor.

6.2 An Effect of our Pruning Rule

In this section, we show an effect of our pruning rule in Phase 1, that is, in the search for finding the candidates for component itemsets. Our experimental result is summarized in Figure 1. In the figure, N is the number of possible itemsets with the probability of non-zero in the local database we are concerned with. That is, it is the size of the whole search space. The computation time for extracting the candidates without the pruning is denoted by t_N. N_{act} is the number of itemsets actually examined in our search with the pruning and $t_{N_{act}}$ is the computation time for the search. N_{cand} denotes the number of the extracted candidates for component itemsets.

The result shows that the number of the candidates to be extracted, N_{cand}, is much smaller than the number of the possible ones in each case (region). Therefore, finding the candidates without any pruning will be quite impractical. As shown in the figure, since the pruning rule can reduce at least 90 % of the whole search space, it can be considered that our pruning can work well to improve the search efficiency in Phase 1.

$\rho = 3.0, \zeta = 0.4$							
region	$P(C)$	ϵ	N	N_{act}	N_{cand}	t_N(sec)	$t_{N_{act}}(sec)$
Atlanta	0.0642	0.0922	2.4×10^7	1.8×10^6	3.5×10^4	144.031	17.906
Boston	0.105	0.118	2.4×10^8	4.5×10^6	4.5×10^4	1428.641	43.985
Chicago	0.163	0.147	4.8×10^7	3.1×10^6	4.7×10^4	283.172	28.735
Los Angeles	0.108	0.118	2.7×10^7	1.8×10^6	1.2×10^4	161.578	17.656
New Orleans	0.0786	0.102	1.5×10^7	1.4×10^6	1.2×10^4	89.656	12.735
New York	0.289	0.196	1.6×10^7	1.5×10^6	7.2×10^4	95.078	14.578
San Francisco	0.0995	0.114	3.0×10^7	2.3×10^6	5.3×10^4	176.141	21.750
Washington DC	0.0940	0.112	2.0×10^9	2.9×10^7	2.2×10^4	11536.375	279.500

Fig. 1. an effect of our pruning rule

region	$\|C_{all}\|$	$\|C_b\|$	$\|C_m\|$	$\|DC\|$	$\|DC_{imp}\|$	$t_{\|C_b\|}$(sec)	$t_{\|C_m\|}(sec)$
Atlanta	6.1×10^8	2.8×10^8	3.0×10^6	1.4×10^6	353	355.922	132.422
Boston	1.0×10^9	4.5×10^8	4.5×10^6	2.9×10^6	240	804.329	223.016
Chicago	1.1×10^9	5.8×10^8	4.5×10^6	3.1×10^6	7	829.906	236.282
Los Angeles	6.7×10^7	4.1×10^7	5.4×10^5	2.5×10^5	101	54.062	16.375
New Orleans	7.4×10^7	4.5×10^7	5.3×10^5	2.2×10^5	57	57.656	20.062
New York	2.6×10^9	1.2×10^9	9.7×10^6	6.8×10^6	44	2820.750	551.547
San Francisco	1.4×10^9	6.0×10^8	4.2×10^6	2.5×10^6	393	900.907	285.953
Washington DC	2.4×10^8	1.3×10^8	9.1×10^5	6.0×10^5	86	216.579	59.079

Fig. 2. an effect of constraint of DC pairs in Phase 2

6.3 An Effect of Constraints of DC Pairs

Let $Comp_{cand}$ be the set of the candidates obtained in Phase 1. In Phase 2, we examine whether a pair of component itemsets in $Comp_{cand}$ can be a DC pair or not. The results for Phase 2 is summarized in Figure 2. In the figure, $|C_{all}|$ is the number of the possible pairs we can extract from $Comp_{cand}$.

$|C_b|$ is the number of pairs of X and Y in $Comp_{cand}$ such that $X \cap Y = \phi$ and $t_{|C_b|}$ is the computation time for finding the DC pairs from C_b. Furthermore, $|C_m|$ is the number of pairs of X and Y in $Comp_{cand}$ such that both of X and Y are contained in a transaction in $\mathcal{D}_\mathcal{C}$. The computation time for finding the DC pairs from C_m is denoted by $t_{|C_m|}$. Finally, $|DC|$ is the number of extracted DC pairs and $|DC_{imp}|$ the number of DC pairs in DC whose degree of correlation is less than or equal to 1.

From the results, by the latter constraint, the number of candidate pairs to be examined can be drastically reduced. Therefore, it is expected that our search in Phase 2 can be performed efficiently. As the result, it is shown that DC pairs can be found efficiently by using our pruning rule and constraint. Further, in the next section, we show our search for DC pairs in this paper is efficient in contrast with our previous search in [1].

region	N	N_{comp}	N_{DC}	$N_{comp} - N_{DC}$	$\lvert C_m\rvert - \lvert DC\rvert$
Atlanta	2.42×10^7	2.36×10^7	4.69×10^5	2.32×10^7	1.64×10^6
Boston	2.42×10^8	2.42×10^8	1.01×10^6	2.41×10^8	1.56×10^6
Chicago	4.78×10^7	4.76×10^7	5.41×10^5	4.70×10^7	1.44×10^6
Los Angeles	2.74×10^7	2.72×10^7	7.56×10^4	2.72×10^7	2.80×10^5
New Orleans	1.51×10^7	1.49×10^7	8.12×10^4	1.49×10^7	3.08×10^5
New York	1.59×10^7	1.56×10^7	8.83×10^5	1.47×10^7	2.95×10^6
San Francisco	2.96×10^7	2.88×10^7	7.90×10^5	2.80×10^7	1.69×10^6
Washington DC	1.95×10^9	1.95×10^9	2.76×10^5	1.95×10^9	3.06×10^5

Fig. 3. a comparison our new method with our previous method

6.4 A Comparison our New Method with our Previous Method

As we discussed in 5.1, we have already tested the way of finding DC pairs that compound itemsets are firstly found in [1]. In order to compare our new method with our previous one, we examine the number of candidates for compound itemsets and whether the candidates can be divided into DC pairs or not. Our experimental result is summarized in Figure 3. In the figure, N is the same number in 6.2, and $|DC|$ and $|C_m|$ are the same number in 6.3. N_{comp} is the number of candidates for compound itemsets in each region. N_{DC} is the number of the candidates which can be divided into DC pairs. Note here that the candidate may be several DC pairs. Although N_{DC} differs from $|DC|$ in 6.3, as a result, the same number of DC pairs can be found by using our previous method.

The experimental result shows that the number of candidates for compound itemsets, N_{comp}, is almost the same number of possible itemsets with the probability of non-zero in the local database, N, in each region. That is, there are a few itemsets which can be pruned even if some pruning rules presented in [1] are used. Therefore, in Phase 1, our new method can be found candidates efficiently in contrast with our previous method.

Although it is difficult to realize efficient search in Phase 1 by using our previous method, this does not mean that the previous method is not efficient if most of the candidates can be divided into DC pairs. However, as we discussed in 5.1, the number of the candidates which can be divided into DC pairs, N_{DC}, is much smaller than N_{comp}, so most of the candidates cannot be divided into DC pairs. Therefore, we have to check many candidates which cannot be DC pairs, further, may examine all subsets of each the candidates in worst case. On the other hand, the number of combinations of candidates for component itemsets which cannot be DC pairs, $|C_m| - |DC|$, is much smaller than the number of the candidates for compound itemsets which cannot be DC pairs, $N_{comp} - N_{DC}$. As a result, by using our new method, we do not have to examine many combinations which cannot be DC pairs.

Thus, it can be considered that our new method realize efficient mining of DC pairs in contrast with our previous method.

6.5 An Example of DC Pair

We have obtained various kinds of DC pairs in the experimental data. For instance, in New Orleans, a DC pair $X = \{$*Entertainment*, *Quirky*, *Up and Coming*$\}$ and $Y = \{$*\$ 15-\$ 30*, *Private Parties*, *Spanish*$\}$ has been found. The pair shows high degree of difference of correlations by conditioning to New Orleans. However, since the pair shows very high degree of correlation in the local database, we will be able to find them as characteristic itemsets by some method previously proposed. On the other hand, we have also found a DC pair, $X = \{$*Quirky*$\}$ and $Y = \{$*Good Decor*, *Italian*, *\$15-\$30*, *Good Service*$\}$ for New Orleans. The pair is not correlated in both global database and local database. Therefore, the pair cannot be found by previous methods. However, the pair shows high degree of difference of correlations by conditioning to New Orleans. Although the correlation of the pair in New Orleans seems to be not so high, it is much higher than one in the global database. We consider such a DC pair can be especially useful in some cases. For instance, people looking for a restaurant in New Orleans may be interested in a "quirky Italian restaurant" which is a hidden feature in New Orleans in contrast with a "quirky Spanish restaurant" which is an explicit feature in the local database because there may be some factor of its high degrees of difference of correlations even if the pair does not show high degree of correlation.

Thus, our algorithm has actually found potentially significant DC pairs for the given database.

7 Concluding Remarks

Given a transaction database $\mathcal{D}$ and its sub-database $\mathcal{D}_\mathcal{C}$, we proposed the notion of DC pairs. A pair of itemsets X and Y is called a DC pair if the correlation between X and Y in $\mathcal{D}_\mathcal{C}$ is relatively high to one in the original $\mathcal{D}$ with some degree. It should be noted that the correlation is not always high in $\mathcal{D}_\mathcal{C}$ even though we can observe some degree of correlation change for $\mathcal{D}$ and $\mathcal{D}_\mathcal{C}$. In this sense, such a pair might not be characteristic in $\mathcal{D}_\mathcal{C}$. Thus, some DC pairs are regarded as *potential characteristics* in $\mathcal{D}_\mathcal{C}$. Our experimental results showed that DC pairs which are potentially significant can be actually found for "Entree Chicago Recommendation Data" under conditioning by each region.

In order to efficiently find DC pairs, we investigated several pruning mechanisms which can prune useless search nodes (branches) and designed an algorithm adopted them. The computation is divided into two phases. In Phase 1, we can efficiently extract the set of candidates for component itemsets with a look ahead strategy. In Phase 2, then, a restricted pairs of obtained candidates are examined whether they can be DC pairs or not. Our experimental results have also shown effectiveness of the pruning rules in our search.

A more powerful pruning mechanism would be desired in more practical cases. We would be able to realize such an improvement of computational efficiency heuristically. For instance, imposing a *semantic constraint* on itemsets will be effective in reducing our search space. We might consider only candidates for compound itemsets each of which contains a certain pair of items semantically interesting. As the result, the number of candidates can be drastically reduced still preserving semantical significance. This kind of constraints will be investigated as future work.

References

1. T. Taniguchi, M. Haraguchi and Y. Okubo. Discovery of hidden correlations in a local transaction database based on differences of correlations. In *Proc. of the IAPR Int'l Conf. on Machine Learning and Data Mining in Pattern Recognition*, 2005 (to appear).
2. R. J. Bayardo. Efficiently mining long patterns from databases. In *Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data, 85-93*, 1998.
3. T. Uno, M. Kiyomi and H. Arimura. LCM ver.2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets. In *Proc. of the IEEE Int'l Conf. on data mining, 2nd Workshop on Frequent Itemset Mining Implementations (FIMI'04)*, 2004.
4. R. Agrawal, R. Srikant. Fast algorithms for mining association rules. In *Proc. of the Int'l Conf. on Very Large Data Bases*, pages 487-99, 1994.
5. G. Dong and J. Li. Efficient mining of emerging patterns: discovering trends and differences. In *Proc. of the 5th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 43-52, 1999.
6. S. D. Bay and M. J. Pazzani. Detecting group differences: mining contrast sets. *Data Mining and Knowledge Discovery*, v 5, n 3, pages 213-46, 2001.
7. G. I. Webb, S. Butler and D. Newlands. On detecting differences between groups. In *Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 256-65, 2003.
8. S. Brin, R. Motwani and C. Silverstein. Beyond market baskets: generalizing association rules to correlations. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, v 26, n 2, pages 265-76, 1997.
9. S. Brin, R. Motwani, J. D. Ullman and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, v 26, n 2, pages 255-64, 1997.
10. C. C. Aggarwal, P. S. Yu. A new framework for itemset generation. In *Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 1998)*, pages 18-24, 1998.
11. S. Morishita and J. Sese. Traversing itemset lattices with statistical metric pruning. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Database Systems (PODS)*. pages 226-36, 2000.
12. Y. Ohsawa and Y. Nara. Understanding internet users on double helical model of chance-discovery process. In *Proc. of the IEEE Int'l Symposium on Intelligent Control*, pages 844-9, 2002.
13. S. Hettich, and S. D. Bay, The UCI KDD Archive, Department of Information and Computer Science, University of California, Irvine, CA, http://kdd.ics.uci.edu, 1999.